EVENT-STREAM SUBSCRIPTION SYSTEMS – COMPARING AND INTEGRATING THE CONCEPTS OF ACTIVE DBMS, EVENT PROCESSING, DATA-STREAM SYSTEMS, AND PUBLISH/SUBSCRIBE SYSTEMS

Meyer-Wegener Klaus, Daum Michael

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Summary

Complex-event processing (CEP) is a popular approach today. The term is often used in combination with data-stream processing and publish/subscribe systems, although these systems have slightly different foci. The root of many techniques used in all of these systems can be found in the literature on active database management systems, which was mostly written in the nineties already. It now seems to be necessary to clarify the differences of these approaches and to identify the application scenarios where each particular system fits best. This article attempts to analyze the different kinds of event processing and to make their differences clear. It concludes with the vision of a unifying approach.

Keywords: Event-processing systems. Data-stream processing. Active database management systems. Publish/subscribe systems.

1. Introduction

The motivation behind event processing in computer systems is twofold: speed (quick reaction, real-time) and expressivity (declarative languages).

Incoming events (usually messages) should not be stored and then later evaluated (e.g. byqueries), but should be processed immediately, on-the-fly. Event processing promises the fastest reactions. However, events very often are not just individual occurrences, but they need to be combined into more complex entities before the decision on some kind of reaction is made. Such reactions can be just display, a warning or alert, or storage (recording). An immediate action with effect on the real world is also possible, but in most applications human control is more desirable.

Second, programming the reaction to incoming events should not be done in standard (imperative) programming languages, but in some kind of declarative language with much more expressivity. One reason is to improve flexibility: defining new reactions should be easy. Secondly, this increases the productivity of the system developer. The third reason is the potential for optimization: a declarative language does not go into the details of the implementation and thus leaves room for choosing an optimal way—which may vary over time.There is a long tradition of language design for reactive systems, mostly coming from interactive systems, i.e. handling user input. The earliest design we are aware of are Dijkstra’sGuarded Commands [1]. Here, the emphasis is on the processing of incoming data elements (messages). As Luckham pointed out in his book on complex-event processing [2], we are dealing with data objects signifying an event.

The philosophical discussion of what an event actually means is not repeated here. There is some literature available on that, see [3] and the list of references in it. The starting point is a data object received by the processing system. This data object always has some type that further describes its structure and its properties, and there is also some additional information on the sequence of objects of the same type, e.g. ordering or frequency. This is necessary for a proper definition of operations on the data objects.A typical assumption in that definition of operators is that the recent past is more relevant than the long-gone past.

The open question remains how these systems should be designed and built. The available approaches are quite different. The presentation starts with a short characterization of the four classes of systems that have already been defined for event processing.

2. System Classes

2.1 Active Database Management Systems - "Trigger"

This is the oldest class of systems with a significant literature in the nineties; see for instance [4, 5]. The specifics here are that events are typically database updates, i.e. inserts, deletes, and modifies. Having in mind that all events are in fact data objects, this is not unrealistic. Usually, a generic event class has also been introduced, together with some kind of "raise" operator that would explicitly state that the particular event has happened. But the impression has always been that this is some kind of second-class event.

The problem of events being database updates is the speed of reaction: Each event must be stored first, only then the processing of events can be triggered. The other classes of systems claim to be much faster. Nevertheless, many problems of event processing have been discussed in much detail already in the context of ADBMS. In particular, the event algebra [5] is quite useful up to date.

And the approach isstill popular today: Oracle’sContinuous Query Notification (CQN)has been used to implement systems that would have been considered as prime candidates for data-stream processing [6].A similar approach for the application of flight control has been taken in [7, 8]. While their processing is for sure not as fast as possible, it is proven to be fast enough—and you do not need a second system!

2.2 Publish/Subscribe Systems - "(Content) Dissemination"

Publish/subscribe is rather a networking paradigm than an event-processing concept. It relates senders and receivers, but that relationship is not fixed, it is based on content instead. Publishers send messages out to the world which are characterized by some metadata (for the whole series of messages as well as for individual messages). Subscribers use these metadata to connect to publishers and to receive individual messages.

Combination of published messages is limited.A publication creates a data object. Subscriptions define queries that select those objects, but a combination of different messages into higher-level events is not in the focus of these systems. Hence, filtering is by far the most important operation.

For a more detailed discussion see for instance[9]. It is based on [10].

2.3 Data-stream Processing - "Sliding Windows"

Data-stream processing is favored by the database community today. There is one important difference with the active DBMS approach: The events, or rather the data objects, are not stored for a longer period of time. The whole set of events that have occurred so far is not available. Instead, sliding windows consisting of relatively small sets of recent events are maintained and evaluated. The benefit of the system is maximized if the windows can reside in the main memory and are never written to secondary storage, i.e. to disks. For the evaluation of the windows, database query languages are reused. The hope is that the enormous experience with database query processing can be transferred to data-stream query processing.

Heterogeneity of the different proposals and prototypes turned out to be a significant problem, in particular since the users are well aware of SQL as a standard. Many SQL dialects have been defined with little differences in the syntax, but huge differences in the semantics. [11] has identified these differences clearly.Apart from the SQL dialects, the other paradigm for the definition of data-stream queries is the use of Boxes-and-arrows diagrams. It may be more intuitive for the users, but certainly adds a moment of procedural definition to the queries. Also, the semantics of the operators drawn as boxes may be just as unclear as they are in the SQL dialects. Windows can be defined per query or per operator. The evaluation can be triggered by the input of new tuples as well as the elapse of time [12, 13].Our own approach called Data Stream Application Manager (DSAM)[14]allows handling the heterogeneity, but it comes at the cost of a rather complex system with partitioning and mapping of global queries.

[15] showed how data-stream processing can be traced back to "append-only" databases, which had been introduced as early as 1992 [16].[17] provides an overview of the similarities that the Aurora data-stream processing system shows with traditional database technologies. That broad overview also surveys active DBMS.

2.4 Complex-event processing- "Temporal Correlation"

Luckham created the notion of complex-event processing (CEP) [2]. It has become very popular in business computing today, see for instance [18]. However, the storage of all events is required. Luckham does not use query languages, but relies on two additional markers of the event data: time and causality. The evaluation is some kind of pattern matching that uses these markers. Complex events can be identified with the help of time, e.g. two events that occur before or after each other, or with the help of causality, i.e. one event occurs as a consequence of some other events having occurred before. Given the relations of the events at one level of abstraction, a complex event at a higher level is defined as an aggregate of the related events.

This is much more general than data-stream processing. The latter can be regarded as a special case of CEP with the data-stream elements representing events. Sensor-data fusion can also be seen as a special application of CEP [19].

The following table summarizes the differences of the four classes of systems:

Table 1: Four Classes of Event-Processing Systems

	Active DBMS	*Pub-Sub*	*DSMS*	*CEP*
Data objects:	updates of stored data	publications (messages)	data items (tuples), messages	(basic, elementary) events
Types:	kinds of DB updates (insert, delete, …)	publishers	many streams (with schema)	event types
Time:	global order		time synchroni-zation	central time, global order
Evaluation, Analysis:	expressions, algebra	subscriptions (filters)	joins (queries)	complex events, expressions
Data base:	event history		streams up to now, windows	event history (or trace)
Latency:	less important	less important	very important	important
Storage:	yes	not needed	no	yes

3. Combined Approaches

Cayuga [20, 21] tries to combine the techniques of publish/subscribe and event processing. It does, however, not try to be a data-stream processing systems and accepts that its query-processing power is limited. The operators like "next" and "fold" are very intuitive and easy to implement with a clear semantics, but they are not very powerful and thus cannot really justify the employment of a generic system.

The desired expressivity of the query language remains to be determined: What kinds of evaluations are really required? The two available benchmarks, namely Linear Road [22] and NEXMark [23], provide the best general definition of requirements. Yet, there is no theoretical definition so far that would be generally accepted.

4. Integration Steps

In the examination of existing event-stream models, we regard the following criteria as relevant for a classification:

- Timestamps and validity.

Most stream-processing systems (SPSs) use timestamps to denote when an event has happened. Timestamps may be generated by the data sources (external timestamps) or by the SPS upon arrival of an event item (internal timestamps). A second timestamp may be added to denote how long an event is valid. Timestamps can be part of either the user data or the metadata. In the first case, queries may access (and possibly even alter) timestamps. In the second case, timestamps can only be used by the SPS internally.

- Uniqueness of items.

Stream models differ in their guarantees for the uniqueness of items. It is usually not possible to guarantee uniqueness of user data (e.g. a sensor node may return the same temperature value several times). If uniqueness of items is required, this has to be done via timestamps. In the absence of timestamps or if timestamps do not guarantee uniqueness (e.g. because of insufficient granularity), a simple sequence of numbers may be used. Uniqueness is usually more of a concern for the mathematically precise definition of semantics than it is for actual query processing.

In addition to stream models, existing SPSs also differ in their delivery semantics.The following criteria may be used to distinguish systems:

- Lost or duplicated items.

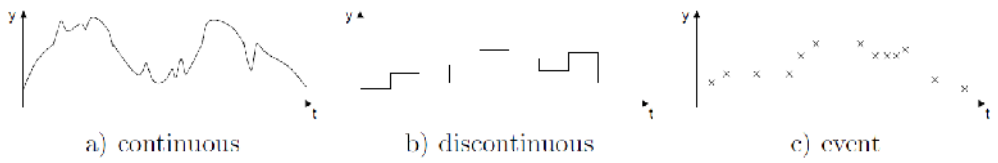
Systems may react differently to lost or duplicated items. Unless there is a notion of a "next" item (e.g. by means of sequence numbers), it is not possible to detect whether tuples have been lost.

- Ordering.

Another interesting difference between SPSs is the issue of ordering. For a data source with either timestamps or sequence numbers, it is possible to wait for out-of-order items and to reorder them. This can be realized by different SPS-specific techniques like sorting with slack parameters, heartbeats, punctuations, or k-constraints. If more than one data source is connected to an SPS, reordering depends on synchronized clocks.

Schmidt [24] distinguishes three kinds of data-stream characteristics as depicted in Fig. 1.

Figure 1: Three Kinds of Data-stream Characteristics [Schmidt2007]



a) continuous:

Continuous values can be turned into a data stream by sampling. A typical example would be temperature measurement or motion tracking. According to the needs of the application, the sampling rate can be adjusted, e.g. to save energy. Interpolation can be useful to estimate values during the time interval of two subsequent measurements.

b) discontinuous:

Discontinuous values often relate to state changes, but not always. An example would be prices for goods. Here, every change should be entered into the system; otherwise false conclusions might be drawn. "Interpolation" is easy because the value remains the same until the next event.

c) (instant) event:

This use of "event" refers to singular appearances of conceptually independent elements. A typical example would be a sales transaction. Nothing important takes place in between two subsequent events.

The kind of event stream given significantly affects the meaning of operations applied to it. Just consider the meaning of something like average. In the case of continuous values, the evaluation can modify the number of events by adjusting the sampling rate. This is not possible for the other two kinds of streams. Operations can also turn one kind of stream into another, e.g. a threshold operator on a continuous-value stream can be used generate a discontinuous-value stream. Sampling on discontinuous-value stream would be an option, but leads to different semantics and most likely creates imprecision, because some of the change events may be missed.

For event-based systems, immediate event processing is appropriate in most cases.Results can be either triggered or they appear periodically.Anintegrated approach must give precise answers the following questions:

- What triggers a result ?
- Which data-stream element or events are relevant for a result ?

In contrast to active databases and publish/subscribe systems, CEP systems and data-stream processing correlate sets of events or data-stream items, respectively.Since it is more than the single event that counts, the precise definition of sets of elements and the appropriate point in time for creating a result are crucial. Data-stream systems use the concept of (virtual) data-stream elements as fixed points that define a sliding window as a continuous segment of the data stream. CEP systems use user-defined descriptions instead that define the set of events and how these events matter for the result.

5. Conclusion

A unifying approach for event-stream processing must regard all aspects of the different models, i.e. the meaning of a data-stream item or event, respectively.In DSAM, we currently use a minimalistic model [14], so our approach is limited to the intersection of the capabilities of our model and the capabilities of the models of the systems that DSAM integrates.In our future work, we will extend our minimalistic model to a model that is as holistic as reasonable.

Query languages should regard all aspects of users' needs.Of course, the expressivity of a query language is limited by the underlying model.Some domains can use temporal windows, while other applications need ordered sequences. The search for simple, yet powerful event-stream model with a declarative, concise query language has only begun.

References:

1. DijkstraE.W. Guarded commands,nondeterminacy and formal derivation of programs. Communications of the ACM, 1975, Vol. 18, No. 8, pp. 453-458.

2. Luckham D. The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Amsterdam: Addison-Wesley Longman, 2002.

3. Lenz R., Schuster H., Wedekind H. Design of (re-)active systems using triggers with complex events. In: Arbeitsberichte,Institut für Mathematische Maschinen und Datenverarbeitung (IMMD), Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, September 1993, Vol. 26, pp. 95-102.

4. Dayal U., Buchmann A.P., Chakravarthy S. The HiPAC project. In: Active Database Systems. Under editionWidom J., Ceri S. Chapter 7.San Francisco: Morgan Kaufmann, 1996, pp. 177-206.

5. Gehani H.H., Jagadish H.V., Shmueli O. Event specification in an active object-oriented database. In: Proc. 2002 ACM SIGMOD Int. Conf. on Management of Data (Madison, Wisconsin, USA, June 3-6). Under editionFranklin M.J., Moon B., Ailamaki, A. ACM, 2002, pp. 81-90.

6. ChandyK.M., Gawlick, D. Event processing using database technology (tutorial). In: Proc. 2007 ACM SIGMOD Int. Conf. on Management of Data (Beijing, China, June 12-14). Under editionChan C.Y., Ooi B.C., Zhou A. ACM, 2007, pp. 1169-1170.

7. Behrend A., Dorau C., Manthey R., Schüller G. Incremental viewbased analysis of stock market data streams. In: Proc. 12th Int. Database Engineering and Applications Symp. (IDEAS, Coimbra, Portugal, Sept. 10-12). Under editionDesai, B.C. ACM, 2008, Vol. 299 of ACM Int. Conf. Proc. Series, pp. 269-275.

8. Behrend A., Dorau C., Manthey R. SQL triggers reacting on time events: An extension proposal. In: Proc. 13th East European Conf. on Advances in Databases and Information Systems (ADBIS, Riga, Latvia, Sept. 7-10). Under editionGrundspenkis J., Morzy T., Vossen G. Springer, 2009, No. 5739 in Lecture Notes in Computer Science, pp. 179-193.

9. Zhou Y., Aberer K., Salehi A., Tan K.-L.Rethinking the Design of Distributed Stream Processing Systems.In: Proc. 24th Int. Conf. Data Engineering Workshops (Cancun, Mexico, NetDB 2008), IEEE Computer Society, 2008, pp. 182-187.

10. CarzanigaA., Wolf A.L. Content-based Networking: A New Communication Infrastructure. In: Proc. NSF Workshop on an Infrastructure for Mobile and Wireless Systems. In conjunction with: Int. Conf. on Computer Communications and Networks (ICCCN. Scottsdale, AZ, Oct.). 2001.

11. Jain N., Mishra S., Srinivasan A., Gehrke J., Widom J., Balakrishnan H., Cetintemel U., Cherniack M., Tibbetts R., Zdonik S. Towards a streaming SQL standard. In: Proc. 34th Int. Conf. on Very Large Data Bases (VLDB, Auckland, New Zealand, August 23-28). VLDB Endowment, 2008, Vol. 1 of Proc. of the VLDB Endowment, pp. 1379-1390.

12. Ghanem T.M., Hammad M.A., Mokbel M.F., Aref W.G., Elmagarmid A.K. Incremental evaluation of sliding-window queries over data streams. IEEE Trans. on Knowl.and Data Eng. 2007, Vol. 19, pp. 57-72.

13. Kopetz H. Event-triggered versus time-triggered real-time systems. In: Proc. Int. Workshop on Operating Systems of the 90s and Beyond (London, UK). Springer-Verlag, 1991, pp. 87-101.

14. Daum M., Lauterwald F., Fischer M., Kiefer M., Meyer-Wegener K. Integration of heterogeneous sensor nodes by data stream management. In: Wireless Sensor Network Technologies for the Information Explosion Era.

Under edition Hara T., Zadorozhny V.I., Buchmann E. Berlin Heidelberg: Springer, 2010, No. 278 in Studies in Computational Intelligence, pp. 139-172.

15. Babcock B., Babu S., Datar M., Motwani R., Widom J. Models and issues in data stream systems (PODS invited talk). In: Proc. 21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS, Madison, Wisconsin, USA, June 3-5). Under editionPopa L. ACM, 2002, pp. 1-16.

16. Terry D., Goldberg D., Nichols D., Oki B. Continuous queries over append-only databases. In: Proc. 1992 ACM SIGMOD Intl. Conf. on Management of Data, June 1992, pp. 321-330.

17. Carney D., Cetintemel U., Cherniack M., Convey C., Lee S., Seidman G., Stonebraker M., Tatbul N., Zdonik S. Monitoring streams - a new class of data management applications. In: Proc. 28th Int. Conf. on Very Large Data Bases (VLDB, Hong Kong, China, August 20-23).Morgan Kaufmann, 2002, pp. 215-226.

18. Mühl G., Fiege L., Pietzuch P. Distributed Event-Based Systems. Springer, 2006.

19. Nakamura E.F., Loureiro A.A.F., Frery A.C. Information fusion for wireless sensor networks: Methods, models, and classifications. ACM Computing Surveys 2007, Vol. 39, No. 3.

20. Demers A., Gehrke J., Hong M., Riedewald M., White W. Towards expressive publish/subscribe systems. In: Advances in Database Technology - Proc. 10th Int. Conf. on Extending Database Technology (EDBT, Munich, Germany, March 26-31). Under edition Ioannidis Y.E., Scholl M.H., Schmidt J.W., Matthes F., Hatzopoulos M., Böhm K., Kemper A., Grust T., Böhm C. Springer, 2006, Vol. 3896 of Lecture Notes in Computer Science, pp. 627-644.

21. Demers A., Gehrke J., Panda B. Cayuga: A general purpose event monitoring system. In: Proc. 3rd Biennial Conf. on Innovative Data Systems Research (CIDR, Asilomar, CA, USA, January 7-10). www.cidrdb.org, 2007, Online Proceedings, pp. 412-422.

22. Arasu A., Cherniack M., Galvez E.F., Maier D., Maskey A., Ryvkina E., Stonebraker M., Tibbetts R. Linear road: A stream data management benchmark. In: Proc. 30th Int. Conf. on Very Large Data Bases (VLDB, Toronto, Canada, August 31 – Sept. 3). Under editionNascimento M.A., Özsu M.T., Kossmann D., Miller R.J., Blakeley J.A., Schiefer K.B. Morgan Kaufmann, 2004, pp. 480-491.

23. Tucker P., Tufte K., Papadimos V., Maier D. NEXMark – A Benchmark for Queries over Data Streams.DRAFT.OGI School of Science & Engineering at OHSU, 2002.URL http://datalab.cs.pdx.edu/niagara/pstream/nexmark.pdf.

24. Schmidt S. Quality-of-Service-Aware Data Stream Processing.Ph.D. thesis, TU Dresden, 2007.

ПОДПИСНЫЕ СИСТЕМЫ ПОТОКА-СОБЫТИЙ - СРАВНЕНИЕ И ИНТЕГРАЦИЯ ПОНЯТИЙ АКТИВНЫХ СУБД, ОБРАБОТКИ СОБЫТИЙ, СИСТЕМ ПОТОКА-ДАННЫХ, И СИСТЕМ ПУБЛИКАЦИИ/ПОДПИСКИ

Меиер-Вегенер К., Даум М.

Университет Ерланген-Нюрнберг, Германия

Резюме

Обработка сложных событий является актуальным подходом на сегодняшний день. Термин часто используется в сочетании с обработкой потока данных и системами публикации/подписки, хотя эти системы имеют немного разные фокусы. Основу многих методов, используемых во всех этих системах можно найти в литературе по активным системам управления базами данных, которые были описаны уже в девяностых годах. Необходимо уточнить различия этих подходов и идентифицировать прикладные сценарии, где каждая конкретная система подходит лучше всего. В этой статье сделана попытка проанализировать различные виды обработки событий и сделать эти различия очевидными. Это проявляется в видении единого подхода.

მოვლენათა ნაკადების ხელმოწერითი სისტემები – აქტიურ მონაცემთა ბაზების, მოვლენათა დამუშავების, მონაცემთა ნაკადის და გამოცემა/ხელმოწერის სისტემების ცნებათა შედარება და ინტეგრაცია

კლაუს მეიერ-ვეგენერი, მიხაილ დაუმი

ერლანგენ-ნიურნბერგის უნივერსიტეტი, გერმანია

რეზიუმე

რთული მოვლენების დამუშავება აქტუალური მიდგომაა დღეისათვის. ტერმინი ხშირად გამოიყენება მონაცემთა ნაკადის დამუშავებასა და პუბლიკაცია/ხელმოწერის სისტემებთან ერთად, თუმცა მათი ფოკუსები განსხვავებულია. ამ სისტემებში გამოყენებადი ბევრი მეთოდის საფუძველი შესაძლოა ვიპოვოთ მონაცემთა ბაზების მართვის აქტიური სისტემების თაობაზე არსებულ ლიტერატურაში, რომლებიც უკვე 90-იან წლებში გაჩნდა. აუცილებელია ამ მიდგომათა სხვაობის დაზუსტება და გამოყენებითი სცენარების იდენტიფიცირება, სად რა კონკრეტული სისტემა უკეთ ესადაგება. გადმოცემულია მოვლენათა დამუშავების სხვადასხვა სახეობის ანალიზისა და ამ სხვაობათა კარავების მცდელობა. ეს ვლინდება ერთიან მიდგომაშიც.